*International Journal of Advanced Engineering Research and Science (IJAERS)* *[Vol-7, Issue-4, Apr- 2020]*
*https://dx.doi.org/10.22161/ijaers.74.56* *ISSN: 2349-6495(P) | 2456-1908(O)*

# Deep Learning Models for Visual Inspection on Automotive Assembling Line

Muriel Mazzetto, Marcelo Teixeira, Érick Rodrigues Oliveira, Dalcimar Casanova

Federal University of Tecnology - Paraná (UTFPR), Paraná, Brazil

***Abstract***— *In automotive manufacturing, assembly tasks depend on visual inspection to ensure product and process quality, for example, scratches identification on machined surfaces or correct part identification and selection, which may belong to more than one type of vehicle to be produced on the same manufacturing line. Typically, these tasks are essentially human-led, which have recently been supplemented by the artificial perception provided by computer vision systems (CVSs). Despite their relevance, the accuracy of CVSs depends mostly on the environment control, providing appropriate lighting, enclosure, and stops for images to be collected. These problems makes the solution expensive and overrides part of its benefits, mainly when it interferes with the operating cycle time. Thus, this paper proposes the use of deep learning-based methodologies to assist in the visual inspection task, generating little influence on the original manufacturing environment and exploring it as an end-to-end tool to ease CVSs setup. The approach has illustrated by four proofs of concept in a real automotive assembly line based on models for object detection, semantic segmentation, and anomaly detection.*

***Keywords*— *multiple object detection · anomaly detection · semantic segmentation · automotive assembly line.***

## I. INTRODUCTION

For many years, there was no compromise between product variety and production quantity. Until 1960, there were 2 basic types of manufacturing systems, one characterized by a high production volume but with low product variation, and the second with an ability to handle product variation but with low production. With the advent of flexible manufacturing systems, a compromise has been struck between the number of parts produced and the degree of variation in the products. The concept of a flexible manufacturing system filled the gap between these two original ends.

In addition to achieving a compromise between product variety and production volume, the flexible manufacturing system has other objectives, including [1]:

- Improved Operational Control: Integrating computer systems into the shop floor enables process control, reducing the need for human communication and providing an infrastructure that reacts quickly to manufacturing plan deviations.
- Direct labour reduction: This is usually achieved by automating existing operation. There are three levels of automation: fixed automation, flexible automation, and programmable automation.
- Improved short-term responsiveness: Short-term responsiveness may be the result of processing changes, material delivery, or other engineering issues.
- Improved long-term responsiveness: A flexible manufacturing system must be designed to respond to long-term changes through variations in product volumes, the addition of new processes or new products on the production line.

All these goals of flexible manufacturing bring complications inherent to the computer systems used. These must be advanced enough that they can quickly adapt to variations or deviations from the original production schedules, identify and distinguish between the different products being processed on the shop floor, perform physical changes of physical configuration and be able to adapt to new models as they are gradually introduced into the line.

We can divide the computer systems employed in flexible manufacturing into some categories. [1] automation systems (e.g. production line robots), which will effectively manipulate component parts, assemble and produce; planning systems (e.g. DES, Petry nets), which are responsible for planning and changing the flow of components and products on the shop floor, adapting to the desired production needs of management or supply and supply problems [2]; and quality control systems (e.g. cameras and sensors), responsible for automatically inspecting possible production failures or defects in the final product.

In this third category of systems, automatic quality inspection, various types of technology can be employed [3]. One of the most common are computer vision systems, which use camera-generated images for automatic (non-human presence) or semi-automatic (human inspection) verification of problems such as compliance or fault detection [4].

However conventional vision systems are strictly dependent on fairly controlled environments. This makes them unsuitable for flexible manufacturing, which demands constant and rapid reconfiguration of the production line to shift from a product to another, or to include new products [5]. In practice, these are severe limitations for CVSs, because they have to quickly deal with texture and luminosity variations, new templates, and dynamic analysis [6]. For a CVSs to work under these circumstances, it would require tuning several parameters within a short time. As this is a manual task, inspection can be poor or unfeasible to be done automatically [7]. Also, the efficiency of a CVS depends on robust lighting control, which implies to stop the production line to fix work pieces for images acquisition. This procedure makes the solution expensive because it increases the operating time over each work piece, the so-called *cycle time*, and reverting part of its benefits.

This difficulty, especially in adapting vision systems to new products, increases the value of end products. Not only would it be difficult and delaying the introduction of new products into the line as it is necessary to wait until the computer vision solution was redesigned to work properly on the production line [6]. Often the vision system can be replaced by a human operator until the new solution is adequate for the task [7].

However in recent years we have made great strides in computer vision systems, especially in systems called end-to-end models, based on Deep Learning. Unlike traditional feature-based engineering computer vision systems, this new approach is based on feature learning [6, 8]. Roughly, these models no longer require highly skilled labour to adapt the computer system to new products or tasks in a flexible manufacturing plant.

Nowadays, applications of deep learning include translation, speech and audio processing, social network analysis, healthcare and visual data processing [9, 10].

This paper aims to present a case study of prototype implementation of end-to-end models in a car production line, presenting its virtual advantages over traditional models of computer vision, both in the quality of solution as its facilitated adaptability. Also, possible applications for automated inspection (compliance checking and fault detection) are designed based on the developed vision systems.

Although the vision systems presented here, as they are prototypes, have not been thoroughly tested to verify their true effectiveness in automated inspection, the innovations, quality and possibilities of end-to-end models alone make such a goal realistically tangible.

## II. FEATURE ENGINEERING X FEATURE LEARNING

Historically, the performance of machine learning methods, and in this category included computer vision methods, depend heavily on the choice of data representation (also called characteristics) to which it is applied [11, 9]. The traditional computational view is usually composed of 4 basics steps [12]: (1) image acquisition; (2) pre-processing (e.g. setting brightness, contrast, colour saturation, etc.); (3) feature extraction (i.e., representing the original data in a lower dimensional space) and; (4) classification (e.g. by contour comparison with a template).

All of the steps listed above have their inherent difficulties, and the solution depends heavily on the problem presented. No traditional computer vision approach to date has been generic enough to succeed in any given situation. Steps (2) and (3) are especially complicated since all the pre-processing adjustments, like lighting, parameters settings, etc. It must be done manually for each case and a different data representation should be made on a case by case basis (i.e. a set of computational methods would be used to identify human faces, but the same set would not be adequate to identify flaws in a machined surface in the industry). This has always been a criticism of the computer vision research area since each problem needed a unique solution, requiring highly specialized knowledge for solution development [11].

In this sense, much of the actual effort to deploy computer vision algorithms is directed to the design of pre-processor and feature extraction pipelines that result in a representation of data that can support effective machine learning [9]. This feature engineering is important, but it requires a lot of work, highly skilled people, and highlights the weakness of current learning algorithms: their inability to automatically extract and organize discriminatory information from data

[6].

Over the past few years, we have observed the emergence of high-tech concepts like deep learning, as well as its adoption by some giant organizations [9, 13]. In addition to the high hierarchical structure, deep learning methods also have other features that set it apart from traditional computer vision methods, highlighting the ability to suppress, or at least dramatically reduce, pre-processing and feature extraction in traditional methods. That is, it is said that such models have feature learning capabilities, no longer requiring any engineering or specialized personnel to perform such a task [8].

Deep learning based methods integrate pre-processing, feature extraction and classification into a single model, so they are also called end-to-end architectures, since in this type system input is pure images, without any pre-processing or feature extraction, and output is the desired information generated by the classifier.

Internally in its structure the features, previously extracted manually, are now learned, a process called feature engineering. This process is accomplished by selecting different kernels or adjusting the parameters through end-to-end optimization [9]. Its deep architecture of neural networks with many hidden layers is essentially composed of multilevel nonlinear operations. It transfers the representation (or characteristic) of each layer from the original input to a more abstract representation in the upper layers to find the inherent complicated structures. For example, features such as edge, corner, outline, and object parts are abstracted layer by layer. These representations of abstracted characteristics are inserted into the classifier layer to perform classification or regression tasks. In general, deep learning is an end-to-end learning structure with minimal human inference, where learning model parameters are trained together (i.e. parameters for pre-processing task, feature extraction task and classification task).

Traditional computer vision systems, in turn, perform feature extraction and model building separately, and each module is built step by step. Handcrafted features are extracted first, turning raw data into a different domain (for example, statistical, frequency, and time domain) to obtain representative information that requires specialized domain knowledge. Feature selection is then performed to improve relevance and reduce false redundancy between features before feeding the machine learning model. Traditional neural network techniques often have shallow structures with a maximum of three layers (for example, input, output, and a hidden layer). Thus, the performance of the constructed model depends not only on the optimization of the adopted algorithms (e.g., multilayer neural network, support vector machine and logistic regression), but is also strongly affected by the handcrafted features. Generally, feature extraction and selection is time consuming and relies heavily on domain knowledge.

Therefore, high-level abstract representation in feature learning makes deep learning more flexible and adaptable to the variety of data. Because data is abstracted, the various data types and sources have no strong influence on the results of the analysis. On the other hand, the deep hierarchical structure in deep learning is easier to model the nonlinear relationship compared to the superficial structure that is considered in traditional machine learning. In the context of the age of big data in flexible manufacturing, the ability to avoid feature engineering is considered a major advantage due to the challenges associated with this process.

## III. COMPUTER VISION APPLICATIONS FOR AUTOMATED MANUFACTURING INSPECTION

This section covers three different applications of deep learning models for quality inspection in an automotive production line. Methods for object detection, semantic segmentation and anomaly detection were discussed. Also, a proof of concept of the chaining of two methodologies was studied, to improve the quality inspection in real time. The experiments were performed in assembly and machining lines of Renault do Brazil, seeking to influence as little as possible in the original environment, especially without changing the cycle time of operations. The experiments are separated according to the methodology used.

## 3.1 Object Detection

In a classification task, an object is assigned to one of the predefined classes. Several different classification tasks can be found in the domain of quality control such as: classification of an image to determine the presence or an absence of a specific component; classification between normal and anomaly configurations; classification of components according to their descriptive features, among others [6, 8, 14].

Deep learning is becoming the methodology of choice for classifying manufacture data available as images. The majority of the available deep learning classifiers use convolutional neural networks with a varying number of convolutional layers followed by fully connected layers. The availability of manufacture data is limited as compared to the natural image datasets, which drove the development of deep learning techniques in the last 5 years. Therefore, many applications of deep learning in manufacture image classification have resorted to techniques meant to alleviate this issue: the transfer learning. The transfer learning strategy, which involves fine tuning of a network pre-trained on a different dataset, has been applied to a variety of tasks of classification.

As mentioned earlier, the principal advantage of end-to-end are no choices need to be made regarding which features should be extracted, that is, the classification is performed directly, without the need for any intermediate step, simply informing the desired input and output. The Figure 1 shows examples of data used in our case study in a task of classify an image as disk brake or disk calliper. The images refer to *ground truth*, i.e. the templates used for supervisory learning to learn to detect objects.

Detection is a task of localizing and pointing out (e.g., using a rectangular box) an object in an image. In manufacture, detection is often an important step in the quality control process, which identifies a component or a region of interest for further classification or segmentation [15].

The most common approach to detection for 2-dimensional data is a 2-phase process that requires training of 2 models. The first phase identifies all candidate regions that may contain the object of interest. The requirement for this phase is high sensitivity and therefore generally produces many false positives [16]. A typical deep learning approach to this is a regression network for bounding box coordinates based on architectures used for classification. [17]. The second phase is simply the classification of the subimages extracted in the previous step. The classification step when using deep learning is usually done using transfer learning.

Another approach to detection is a single-phase detector that eliminates the first phase of region proposals. Examples of popular methods that were rst developed for detection in natural images and rely on this approach are You Only Look Once (YOLO) [18], Single Shot MultiBox Detector (SSD) [19] and RetinaNet [20].

The end-to-end model simplifies all the develop process, since you only have to enter the desired input and output. Figure 1 exemplifies 3 input images with desired delimitation for object detection (i.e. ground truth) for the case study presented in this paper. The ground truth consists only of presenting the rectangle in the position where the piece is and the respective class.

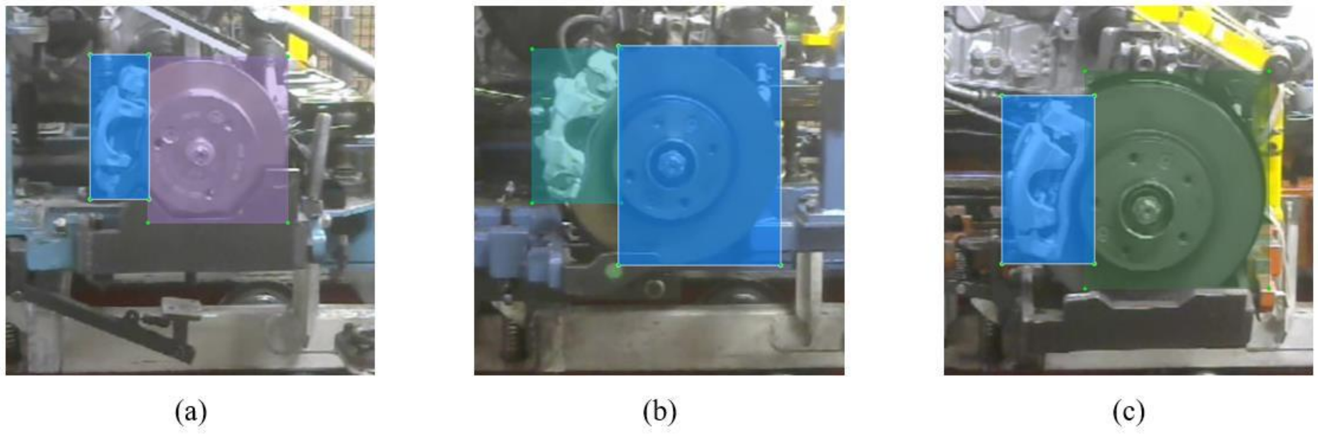


*Fig.1: Ground truth for brake kit object detection. (a) Brake Calliper and Disc 1. (b) Brake Calliper and Disc 2. (c) Brake Calliper and Disc 3.*

Two distinct tasks, brake calliper identification and brake disc identification were performed using the same architecture. The goal was, using a rectangular box, to define the set of pixels that make up each component, a task that tries to make the following tasks easier by selecting only the regions of interest needed (e.g. brake calliper) and eliminating regions where one has full confidence.

The CVSs proposed in this paper uses an *Application Programming Interface* (API) for object detection. The API is integrated to the *Tensorflow* framework [21] and it focuses on measuring and comparing different object detection architectures concerning memory usage, processing speed, and accuracy [22]. The following subsections detail the architecture used, the case study environment and the results.

### 3.1.1 Object Detection Architecture

As object detection architecture, this paper integrates *Single Shot Detection* (SSD) [19] and *MobileNet* [23]. MobileNet acts as a convolution network to extract features from image and the SSD is responsible for objects scanning [22]. These choices are justified mainly because SSD and MobileNet are suitable for projects involving hardware limitations, such as in real-time mobile detection applications [22].

As the proposed architecture is based on CNNs and supervised machine learning, the learning process consists of initially inserting an input image into the CNN, conducting all the convolution and feature extraction operations, until the stage of object detection. Then the error rate is calculated concerning the training set, also called *loss*. The loss value is used to recalculate the CNN weights and the convolution filters, this time following the opposite flow of the architecture.

Every detection is also associated with a *probability* that refers to the percentage of certainty for a model to detect an object. A threshold can be defined for the algorithm to mark a detection only in cases within a specific certainty.

Under the architecture used here, the SSD designs bounding boxes of different sizes and shapes throughout the image. In classification, this architecture assigns a percentage of detection according to the presence of objects in the bounding boxes, and adjusts their respective sizes and positions to match the layout of the identified objects. At the end of the network, a *Non Maximum Suppression* (NMS) algorithm is applied to eliminate bounding boxes redundancies on a same object. The seminal work [19] uses the CNN *VGG-16* [24] as the basis for characteristics extraction. However, the API provided by [22] disintegrates SSD and *VGG-16*, so that other architectures can be combined. Figure 2 shows the architecture of the algorithm proposed in [19] to analyse images with resolution 300 × 300.

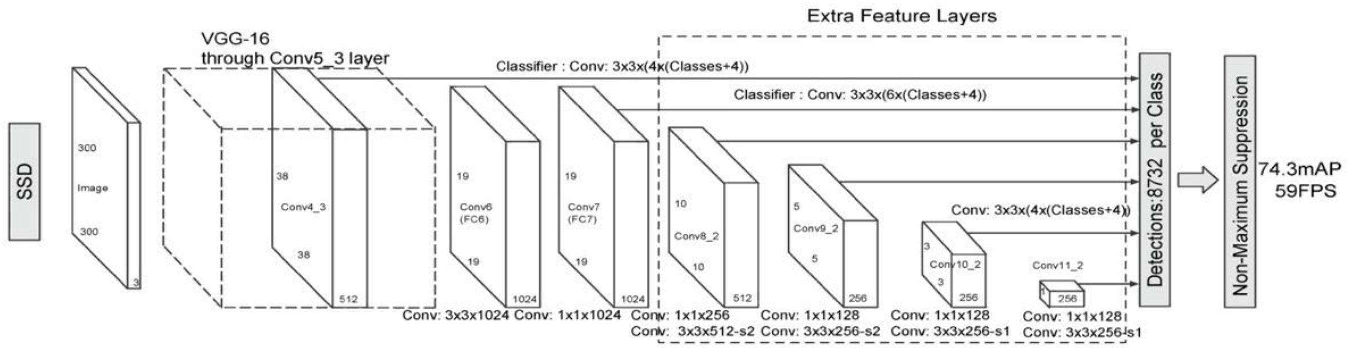


*Fig.2: SSD 300 architecture.*

The foundation for feature extraction in this implementation, the *MobileNet* [23] architecture, replaced *VGG-16* by reducing in 1/30 the computational cost and model size [22]. This architecture is composed by *depthwise separable convolutions*, i.e., it splits a standard convolution into convolution filters for each input channel, and then it applies a convolution 1 × 1, called a *pointwise convolution*, to match the output of the *depthwise* convolution.

The efficiency of the proposed object detection model is now assessed. We use the *Intersection over Union* (IoU) rate to express whether or not a detection is correct, depending on the bounding box location concerning the image [22]. The IoU is calculated as in Eq. 1, where *AD* is the box area detected by the trained model; *AG* is the box area marked by the training set; and *IA* is the intersection area between *AD* and *AG*. The closer to 1, the better is the object detection.

$$IoU = \frac{IA}{(AD + AG - IA)} \tag{1}$$

It is also important to define the probability value from which detection will be accepted, as this influences the value of *AD*.

However, by itself this metric does not consider cases of incorrect classification or non-classification at all. Therefore, to validate the detection result after its integration to the production line, i.e., assessing the efficiency of trained models, the indexes of *Precision* and *Recall* were used.

*Precision* (Eq. 2) determines how correct the model is in detecting and classifying objects, while *Recall* (Eq. 3) determines the relation between success and fail of the classification. These two metrics are derived from the number of *True Positive* (TP), *False Positive* (FP) and *False Negative* (FN), such that:

- TP is every correct classification for which the bounding box has a IoU more than 0.5;
- FP combines: detections with IoU less or equal to 0.5; detection of classes that are not part of the original image; or overlapping detection, which has already been counted as TP;
- FN is all unrealized classification, i.e., images for which the model was unable to mark the bounding box correctly.

$$Precision = \frac{TP}{(TP + FP)} \quad (2)$$

$$Recall = \frac{TP}{(TP + FN)} \quad (3)$$

Using the concepts presented so far, the next subsection discusses the application of the multiple object detection model to the automotive industry.

### 3.1.2 Object Detection Case Study

The following experiment was conducted on a real automotive assembly line, at the Renault do Brazil vehicle factory. The selected case concerns to the brake disc and calliper set conformity and anomaly detection.

In the assembly line used in this experiment, *Automatic Guided Vehicles* (AGVs) are loaded with workpieces for all types of products to be manufactured throughout the line. As humans conduct this task, it is error-prone, and kits can be assembled with components belonging to different vehicle models. To mitigate this problem, we propose here a detection system that verifies whether or not the kit is assembled only with appropriate workpieces. Test were performed with three types of brake disc and three types of brake calliper, totalling six classes to be learned and analysed by the deep learning model. Figure 3 shows examples of sets and their respective classes.

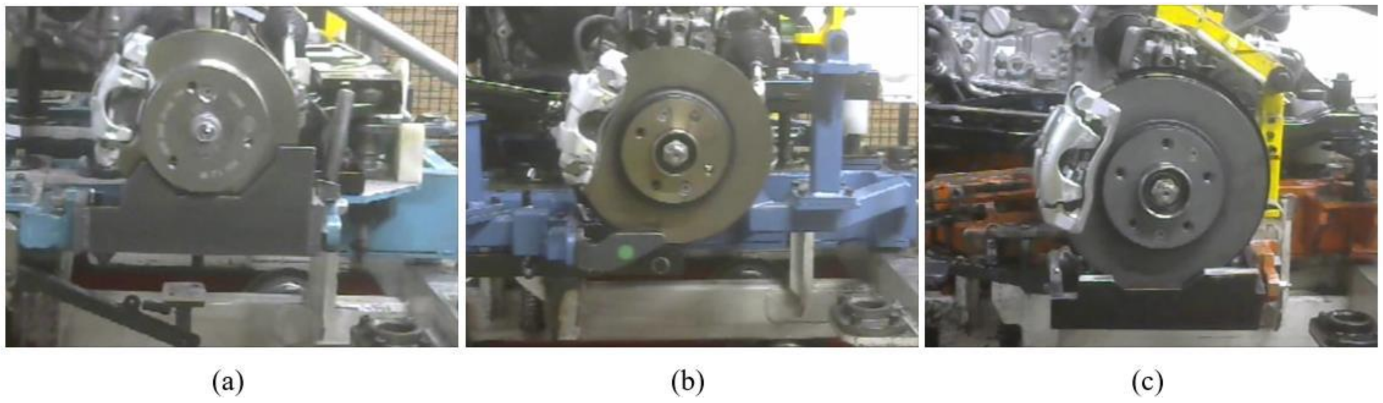


*Fig.3: Types of brake disc and calliper mounted on track support. (a) Brake calliper and Disc 1. (b) Brake calliper and Disc 2. (c) Brake calliper and Disc 3.*

Because it is a factory environment in which each class is standardized, that is, there are no variations in shape or colour between pieces of the same type, it was empirically chosen to use few images for training the object detection model. We used 20 images of each class for the training set, with resolutions of 400 × 400 pixels. For the set of tests, we used

15 images of each type, also with resolutions of 400 × 400 pixels. The choice for square images was due to the resizing performed by the SSD architecture.

The following subsection present the results to this trained model.

### 3.1.3 Object Detection Results

To test the model, we used 321 new images collected from video frames captured on the assembly line. Table 1 presents the results for six evaluated classes, with probability threshold of 90%. The threshold values used for the detection probability were 60%, 65%, 70%, 75%, 80%, 85%, 90% and 95%, making it possible to analyse *Precision* and *Recall* rates, depending on the certainty of detection. Figure 5 shows *Precision* and *Recall* among the analysed classes, to the probability threshold adopted for detection.

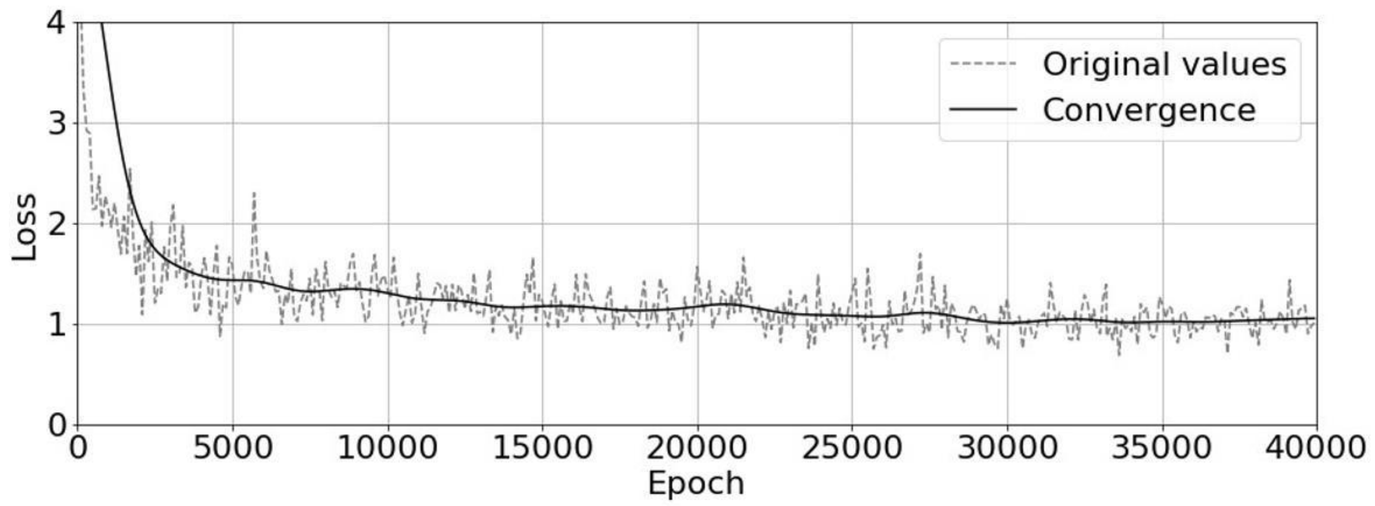
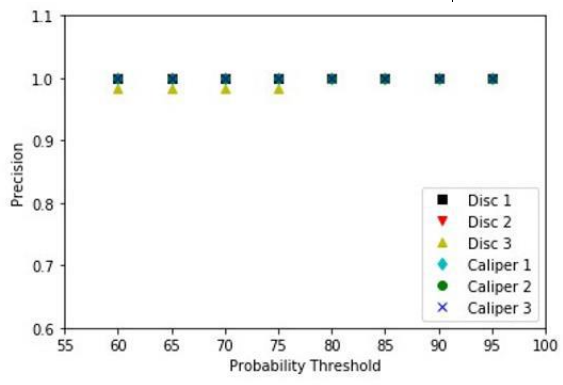


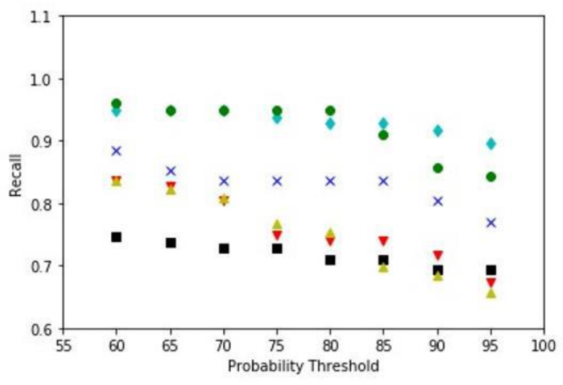
*Fig.4: Learning convergence (i.e., loss as a function of steps) for brake disc and calliper detection model.*

Table 1: Disc and calliper detection in images

| | Disc | | | Calliper | | |
|---|---|---|---|---|---|---|
| Type | 1 | 2 | 3 | 1 | 2 | 3 |
| TP | 79 | 66 | 50 | 89 | 66 | 49 |
| FP | 0 | 0 | 0 | 0 | 0 | 0 |
| FN | 35 | 26 | 23 | 8 | 11 | 12 |
| *Precision* | 1.00 | 1.00 | 1.00 | 1.00 | 1.00 | 1.00 |
| *Recall* | 0.69 | 0.72 | 0.68 | 0.91 | 0.86 | 0.80 |



(a)



(b)

*Fig.5: (a) Precision × Probability threshold. (b) Recall × Probability threshold.*

Note that in Figure 5a, the higher is the detection probability threshold, the higher is the *TP* rate of the model for all analysed classes, due to the low standard deviation. In contrast, by Figure 5 b, also the higher is the amount of *FN*, i.e., of unclassified work pieces.

By analysing the cases that generated *FN*, we found out that they were related with images hiding part of the unclassified component, due to improper video capture, as illustrated in Figure 6.

Summarizing, the approach used to detect images from video frames adds numerous benefits to the assembly line. Among them, the possibility of detecting objects without changing or stopping the production line. A side effect is, however, capturing frames at the beginning or the of the work piece's path, as in Figure 6.

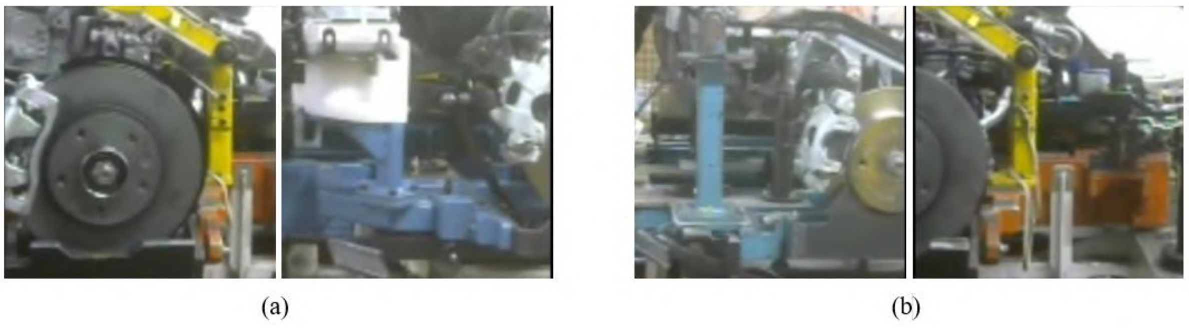


*Fig.6: Samples of unclassified images, which generated FN. (a) Model was unable to classify the calliper. (b) Model was unable to classify the disc.*

However, this effect can be easily outlined by applying temporal configuration for the capture system. For this article, a trigger has been configured to select classes during the precise time work pieces pass through the camera. Thus, the final capture can be based on the classes that were identified in more frames during that period, chosen, for example, by majority vote.

To test this proposal, new videos were captured with 30 frames per second and with resolution of 720 × 720 pixels. The analysed video segment refers to the period within which the work pieces pass through the camera, taking the first successful detection as the initial trigger, and stopping it after 100 frames without any detection.

Figure 7 displays excerpts from one of the tests, containing information such as the total frame count as the part passes the camera's field of view, the object count, and the class detected at the end of the frame count. Figure 7 a displays the frame of the first detection, which starts counting. Figure 7 b displays when the kit is midway in front of the camera's field of view. Figure 7 c displays a frame with FN in brake disc detection. Lastly, Figure 7 d displays the end of the count after 100 consecutive frames without detection.

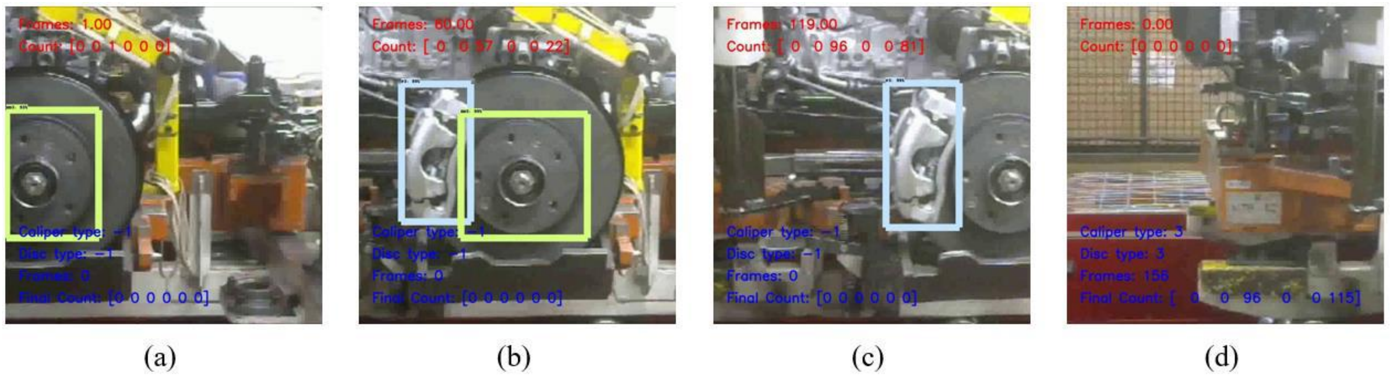


*Fig.7: Counting detection of brake disc and calliper mounted on track support. (a) Start of count. (b) Half of the detection path. (c) Frame with FN in brake disc detection. (d) End of count.*

Three videos were used to evaluate each set of assembled disc and calliper, totalling nine videos to test the entire proposal. Table 2 shows the amount of classes for each video, considering a 95% probability threshold for detection. In this table, videos 1-3 were assembled with disc and calliper type 1; videos 4-6 have type-2; and videos 7-9 have the set assembled with type 3. The amount of frames per video varied with the progress of the production line.

Observing the values in Table 2, one can see that it has been achieved a 100% detection accuracy. Therefore, this approach allows to mitigate the *FN* problem and incorrect detection, as it does not binds the final decision to a single image, but the majority vote within the analysed video segment.

Although it is a good tool for locating objects and classifying them, it is not yet able to classify the boundaries of each object analysed, nor even categorize different regions of the same object. With this problem in mind, a proof of concept with semantic segmentation was performed, addressed in the next section.

*Table 2: Disc and calliper detection in videos*

| | Disc type: | | | Caliper type | | |
|---|---|---|---|---|---|---|
| Video | 1 | 2 | 3 | 1 | 2 | 3 |
| *1* | 113 | 0 | 0 | 119 | 0 | 0 |
| *2* | 105 | 0 | 0 | 118 | 0 | 0 |
| *3* | 103 | 0 | 0 | 103 | 0 | 0 |
| *4* | 0 | 118 | 0 | 0 | 115 | 0 |
| *5* | 0 | 117 | 0 | 0 | 116 | 0 |
| *6* | 0 | 115 | 0 | 0 | 112 | 0 |
| *7* | 0 | 0 | 96 | 0 | 0 | 115 |
| *8* | 0 | 0 | 112 | 0 | 0 | 120 |
| *9* | 0 | 0 | 108 | 0 | 0 | 119 |

## 3.2 Semantic Segmentation

In an image segmentation task, an image is divided into different regions to separate distinct parts or objects cite [25]. In manufacture, the common applications are segmentation of parts often as a pre-processing step for feature extraction and classification [26].

The most straightforward and still widely used method for image segmentation is classification of individual pixels based on small image patches (both 2-dimensional and 3-dimensional) extracted around the classified pixel. It allows for using the same network architectures and solutions that are known to work well for classification, however, there are some shortcomings of this method [25]: the first one is its computationally inefficiency, since it processes overlapping parts of images multiple times; the another drawback is that each pixel is segmented based on a limited-size context window and ignores the wider context. In some cases, a piece of global information, e.g. pixel location or relative position to other image parts, may be needed to correctly assign its label.

One approach that addresses the shortcomings of the pixel-based segmentation is a fully convolutional neural network

(fCNN) [27, 28, 29]. Networks of this type process the entire image (or large portions of it) at the same time and output a 2-dimensional map of labels (i.e., a segmentation map) instead of a label for a single pixel. Example architectures that were successfully used in both natural images and radiology applications are encoder-decoder architectures [30, 31].

Also as in other tasks, the end-to-end model simplifies all the segmentation process. The Figure 11 exemplifies an input image and the desired output image in a task were a cylinder head image need to be segmented to identify machined regions. Inspection of machined regions is especially important as it must be free of defects which, after mounting the engine, could result in loss of power, leakage and engine malfunction.

This paper uses a tensorflow API for semantic segmentation [32], called Deeplab V3+, that combines the benefits of spatial pyramid pooling module and encode-decode structure to increase segmentation task results [26]. The following subsections detail the architecture used, the case study environment and the results.

### 3.2.1 Semantic Segmentation Architecture

The architecture for semantic segmentation used in this paper is the Deeplab V3+, that applies several parallel atrous convolution with different rates, called Spatial Pyramid Pooling (ASPP), to capture the contextual information at multiple scales [26].

The Deeplab V3+ architecture employs the spatial pyramid pooling module (Figure 8 a), with the encoder-decoder structure (Figure 8 b). By that, DeepLabv3+ contains rich semantic information from the encoder module, while the detailed object boundaries are recovered by the simple yet effective decoder module. The encoder module allows to extract features at an arbitrary resolution by applying atrous convolution (Figure 8 c).

The DeepLabv3+ employs an encoder-decoder structure. The encoder module encodes multi-scale contextual information by applying atrous convolution at multiple scales, while the simple yet effective decoder module refines the segmentation results along object boundaries. The Figure 9 shows the details of the encoder-decoder structure within an example.

Differently from the conventional 3×3 depthwise separable convolution (Figure 10 a), that apply a single filter for each input channel, alongside a pointwise convolution (Figure 10 b), that combines the outputs from depthwise convolution across channels, the atrous separable convolution with rate = 2 (Figure 10 c) takes pixel information by adding an interval of one pixel in the stride size. By changing the rate, it is possible to explicitly control the resolution of features computed by deep convolutional neural networks and adjust filter's field-of-view to capture multi-scale information [26]. Note that a standard convolution is the same as an atrous convolution with rate = 1.

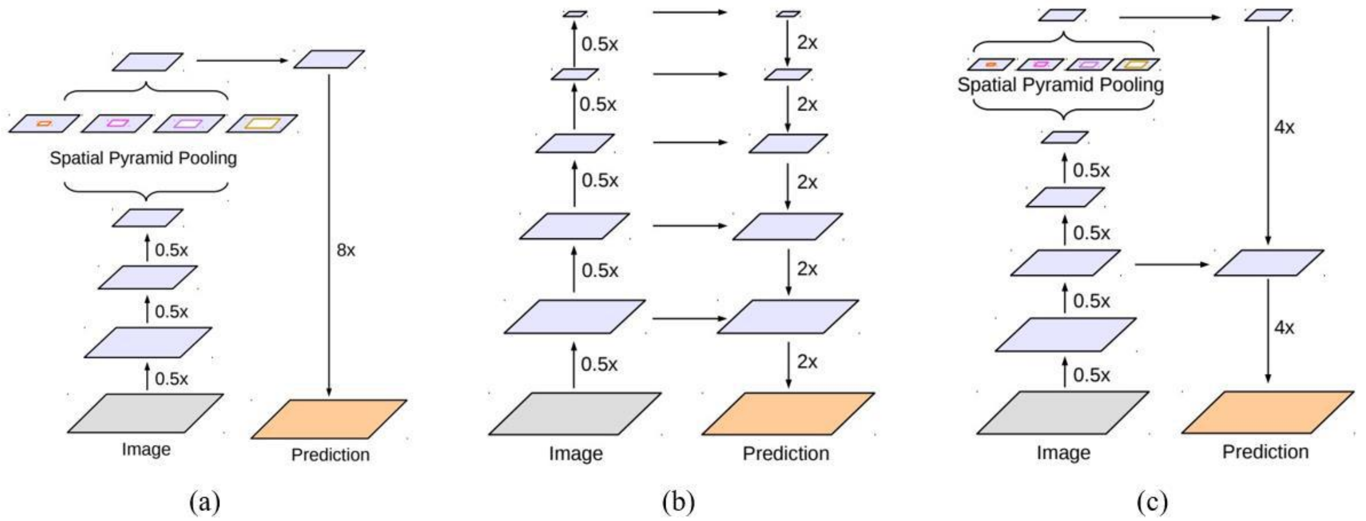


*Fig.8: Deeplab V3+ components representation [26]. (a) Spatial Pyramid Pooling; (b) Encoder-Decoder; (c) Encoder-Decoder with Atrous Conv.*

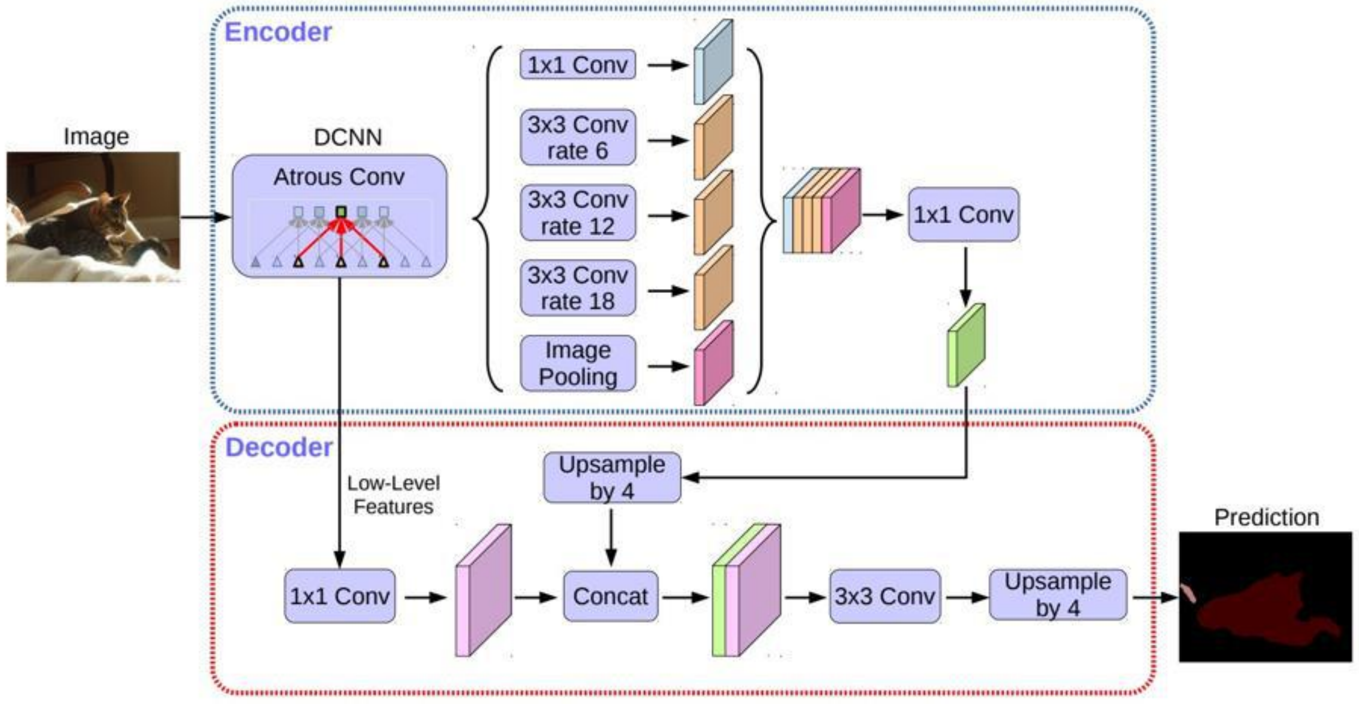


*Fig.9: Encoder-Decoder structure [26].*

Because it is a supervised learning model, each image must have an equivalent label map. The creation of this map consists in colouring specific regions of the image with pre-established colours that represent the classes present in the image. This article used the GIMP tool to generate the label map of the images.

The model evaluation is performed by extending the IoU (Eq. 1), presented in section 3.1, to all the pixels on the image. The API already provides an evaluating tool based on the IoU [32].

The next section makes use of the concepts covered here, applied in a proof of concept to segment parts regions in a machining line.

### 3.2.2 Semantic Segmentation Case Study

The case study approached with semantic segmentation, using the Deeplab V3+ API [32], was carried out in a real cylinder head machining line at the Renault do Brazil engine factory. This methodology was addressed to generate masks to evaluate part quality at the end of the production line, to help the operator focus on performing the inspection task. Model training was performed using a *GeForce GT 540m GPU* with 2 Gigabytes of memory.

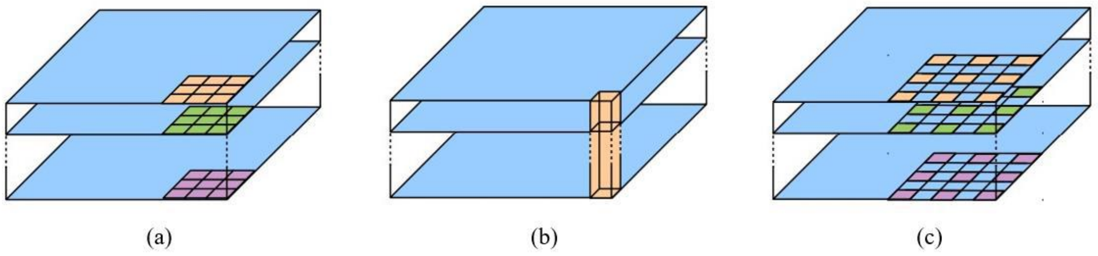


*Fig.10: Example of kernel's dimensions for convolution [26]. (a) Depthwise convolution; (b) Pointwise convolution; (c) Atrous convolution.*

The test performed is just a proof of concept (PoC), without focusing on maximizing hit rates or optimizing training. The general objective of this study is to verify the possibility of using a deep learning based semantic segmentation system as an aid to the inspection task.

The proposed system consists in capturing the image of the face of a piece and generating masks in different regions. Moreover, in the proposed proof of concept, the model was trained with samples of defective parts to generate suggestions of problem regions. Figure 11 shows an example of a cylinder head on the inspection table next to its respective label map.

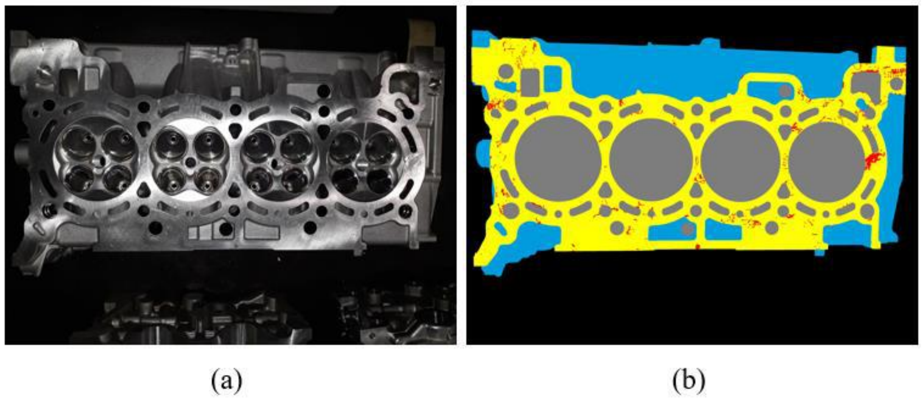


*Fig.11: Example of cylinder head images used in training dataset. (a) Cylinder head on the evaluation table. (b) Cylinder head label map.*

In Figure 11b you can observe the following labels: black is used to label anything that does not belong to the cylinder head as background; the blue colour symbolizes regions of the gross cylinder head, i.e. without machining process; the yellow colour represents the machined surface of the cylinder head; grey is used to label the holes in the part; and finally, red was used to label defects in the part.

It is important to note that the defects presented were generated manually in scrap parts, not being part of the actual processes of the current production line, being generated only for tests in the proof of concept presented.

For this experiment, 10 manually catalogued images were used, all of them labelled with GIMP. The images were captured with a smartphone, with dimensions 4128 × 3096 pixels. Eight images were selected to compose the training dataset and 2 to compose the test dataset. To train the system to absorb the details of the images, losing as little information as possible by resizing to the input dimensions, each image was divided into 36 parts, resulting in images with dimensions of 688 × 516 pixels. Therefore, were used 288 images for training and 72 for testing. The model trained for 1000 steps with a size 2 mini-batch, due to the restrictions of the video memory used.

Regarding the architecture parameters, the atrous rates were 6, 12 and 18. The output stride was set as 16, and the decoder output stride as 4. The input images were resized to 321 × 321 pixels, to match with the input from Deeplab V3+ architecture. The base architecture selected to feature extraction is Xception 65.

Thus, the algorithm flow consists of receiving an input image, dividing it into 36 images, evaluating each partial image and reconstructing the final image with the output of each semantic segmentation. The results of this experiment are shown below.

### 3.2.3 Semantic Segmentation Results

As mentioned earlier, the cylinder head images were evaluated at the end of the production line. Images were captured after removing the parts from the conveyor, over a table where they are checked by an operator to catalogue them as good or bad parts.

The face selected for analysis is considered the most important because it is the contact face with the engine block gasket and cannot contain scratches, porosity or excess material as they may cause leakage or obstruction of cooling and lubrication. Besides, it is also the face that has the most contact with the conveyor during fabrication, which may result in scratches of chip rubbing.

The proposed CVS PoC aims to assist the operator by segmenting the regions to be evaluated according to an evaluation sequence, masking the other regions in each step. Besides, the system also aims to present possible regions with defects as a suggestion for the operator.

To test the system 72 respective images were evaluated, respective to two cylinder heads. Figure 12 displays examples of the images evaluated, with their respective label maps and segmentation predictions.

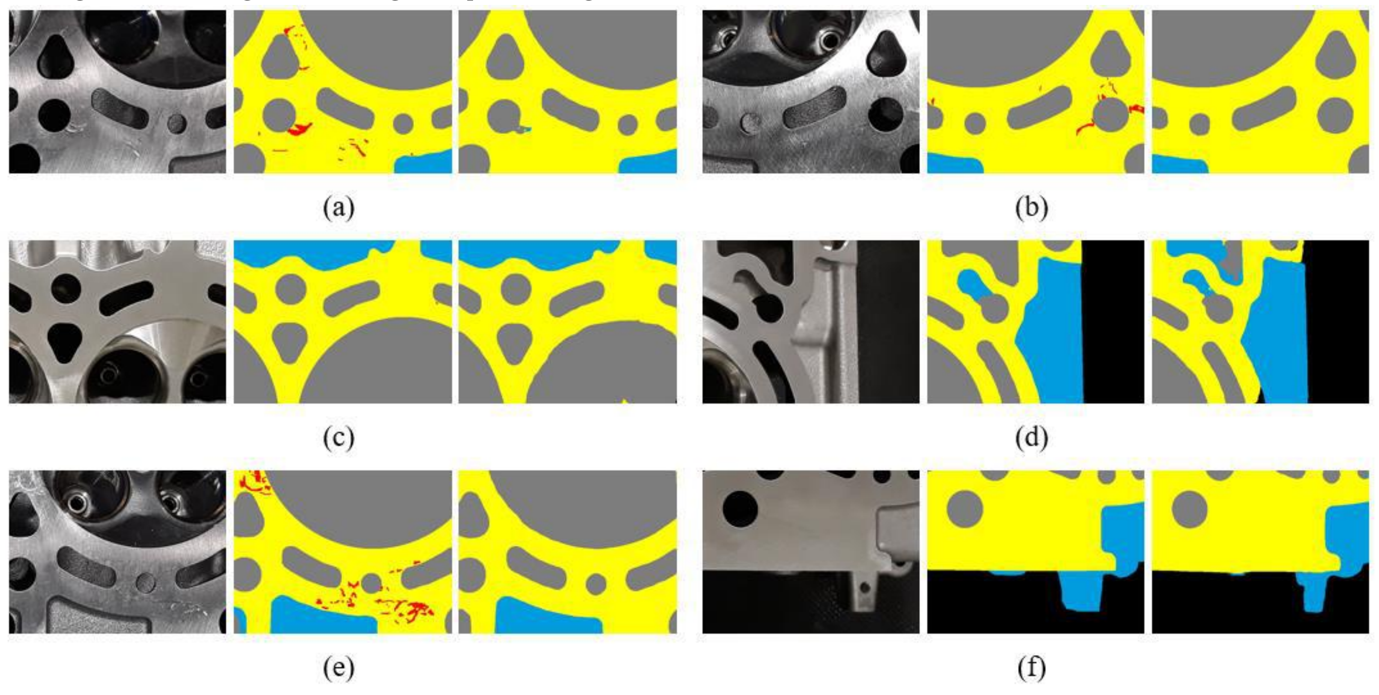


*Fig.12: Original input, label map and semantic segmentation prediction from respective images.*

Observing Figure 12 notably, the trained model could not detect defects. This problem may be related to the low amount of images used in system training, making it impossible to generalize, and also due to the images resizing for model input. In a continuation of PoC, input images must be treated differently, allowing cropping to be made directly at the architectural input dimensions to take advantage of all pixels of the original image. However, it is still possible to verify that the system can create masks on the parts in an acceptable way, following very well the delimitation of the specified regions.

In addition to visual evaluation, the API provides a *mean Intersection over Union* (mIoU) result based on all test dataset images, taking into account the pixels of the semantic segmentation output relative to the label map. For the proof of concept presented in this article, the result of mIoU was 0.7894.

Although requiring the manual creation of label maps, and being a laborious function, the proposed system does not require an expert or engineer to follow the work, it is a task that can be performed by someone who can colorize images with a mouse and keyboard.

To address the problem with defect detection, the following anomaly detection case study was approached.

### 3.3 Anomaly Detection

A common need when you analysing real-world data-sets is determining which data point stand out as being different to all others data points. Such data points are known as anomalies, and the goal of anomaly detection (also referred as outliers, novelties, noise, exceptions and deviations) is to determine all such data points in a data-driven fashion. Errors in the data can cause anomalies but sometimes are indicative of a new, previously unknown, underlying process.

Anomaly detection is a technique used to identify unusual patterns that do not conform to expected behaviour. It can be considered the thoughtful process of determining what is normal and what is not. Anomaly detection is applicable in a variety of domains such as, intrusion detection, health monitoring system in the hospital, fraud detection in credit card transactions in banks, fault detection in operating environments, detection of fake news and misinformation in the Internet, industry quality control inspection, security and surveillance.

The major difficulty in this type of computer vision task is that is nearly impossible to generate a balanced database to train a supervised learning algorithm [14] since anomalous or faulty events are, sometimes, rare to happen. In this way semi-supervised and unsupervised approaches have been dominant recently, as the weakness of supervised approaches is that they require monumental effort in labelling data. On the contrary, semi-supervised and unsupervised methods do not require many data labelling, making them desirable, especially for rare/unseen anomalous cases.

Generative Adversarial Network (GAN) architecture are the most prominent method in this class, especially in computer vision. Unlike traditional classification methods, the GAN-trained discriminator learns to detect false from real in an unsupervised fashion, making GAN an attractive unsupervised machine learning technique for anomaly detection [33]. Also, the GAN framework produces a generator which is an explicit model of the target system with its ability to output normal samples from a certain latent space.

Using a GAN architecture this paper presents two case studies for anomaly detection. One focused only on detecting cylinder head defects in the engine machining line, and the other interested in evaluating the object detection chaining with brake disk anomaly detection. The architecture used, case studies and their results are described in the following subsections.

While in deep learning classification methods (e.g. CNN) input image and its label are required, the anomaly detection GAN based methods only needs the input images. This make this models, probably, the most representative in the class of end-to-end to do learning.

#### 3.3.1 Anomaly Detection Architecture

As anomaly detection architecture, this paper uses a state of art deep learning model called *AnoGAN*, chosen by interest in using recent deep learning methods [34] and the good results in benchmarks presented by [14].

The architecture used in this article is based on GANs, a framework proposed by [34] that consists of two adversarial modules, a generator $G$ and a discriminator $D$. The generator $G$ learns a distribution $p_g$ over data $x$ via a mapping $G(z)$ of samples $z$, 1D vectors of uniformly distributed input noise sampled from latent space $Z$, to 2D images in the image space manifold $X$, which is populated by healthy examples. In this setting, the network architecture of the generator $G$ is equivalent to a convolutional decoder that utilizes a stack of strided convolutions. The discriminator $D$ is a standard CNN that maps a 2D image to a single scalar value $D(\cdot)$. The discriminator output $D(\cdot)$ can be interpreted as probability that the given input to the discriminator $D$ was a real image $x$

sampled from training data $X$ or generated $G(z)$ by the generator $G$ [14]. In other words, this method trains a generative model and a discriminator to distinguish real and generated data simultaneously, illustrated by Figure 13.

Having a trained generator $G$ and discriminator $D$, given a new query image $x$, the function $\mu(x) = x \rightarrow z$ proposed by [14] aim to find a point $z$ in the latent space that corresponds to an image $G(z)$ that is visually most similar to query image $x$ and that is located on the manifold $X$, that is, it seeks to recreate an image considered normal from the characteristics found in the query image $x$.

The anomaly detection score $A(x)$ (Eq. 4) consists of a *residual loss* $L_R(z)$ (Eq. 5), that measures the visual dissimilarity between query image $x$ and generated image $G(z)$ in the image space, and a *discrimination loss* $L_D(z)$ (Eq. 6) calculated by imputing the generated image into the discriminator model, where $\sigma$ is the sigmoid cross entropy with logits $D(G(z))$ and targets $\alpha = 1$.

$$A(x) = (1 - \lambda) \cdot L_R(\mu(x)) + \lambda \cdot L_D(\mu(x)). \tag{4}$$

$$L_R(z) = |x - G(z)|. \tag{5}$$

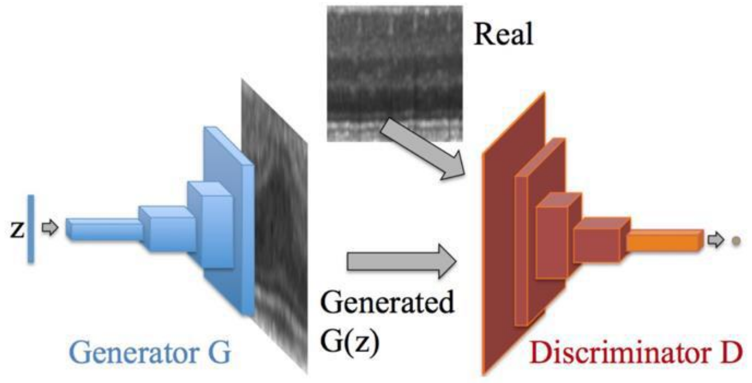


*Fig.13: Deep convolutional generative adversarial network [14].*

$$L_D(z) = \sigma(D(G(z)),\alpha). \tag{6}$$

Additionally, the residual image described by Eq. 7 is used to generate the color map for identification of possible anomalous regions within an image. The colormap selected to this work is the colormap jet, availeble on opencv.

$$x_R = |x - G(z)|. \tag{7}$$

The architecture used for anomaly detection is based on the Adam optimizer, with a learning rate of 0.00001 and $\beta = 0.1$ for the discriminator model and a learning rate of 0.0002 and $\beta = 0.5$ for generator model, choices based in [14]. The latent space vector dimension has been set to 100.

The input is set to receive RGB images of 200 × 200 pixels. The sequence of anomaly detection process is illustrated in

Figure 14, where *model_1* refers to discriminator model and *model_2* refers to generator model. Figure 15 and Figure 16 shows the architectures of discriminator and generator models, respectively.

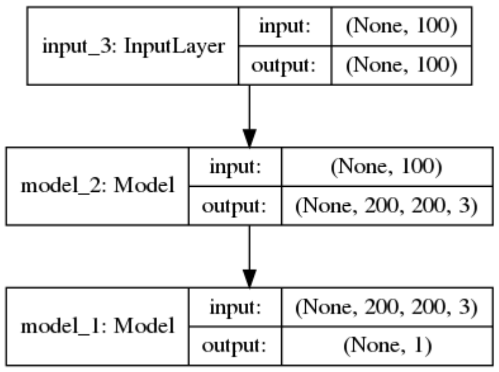


*Fig.14: Anomaly detection process.*

The same training concepts covered in section 3.1 are valid for AnoGAN training aspects, except that it is an unsupervised learning algorithm.

### 3.3.2 Cylinder Head Case Study

The first case study was carried out on a cylinder head machining line, evaluating one of the part faces at the end of the production line. The face selected for analysis is the same mentioned in the semantic segmentation case study.

Although the factory environment has constant artificial lighting, the incidence of external light through the windows varies according to the day and the period. Therefore, solutions proposed with conventional CVSs require enclosure of the part, depriving of external light, and dedicated illumination to highlight imperfections, requiring costly changes in the conventional plant environment. Also, as it is a specifically designed lighting system, slight variations in part angle result in a lot of reflection.

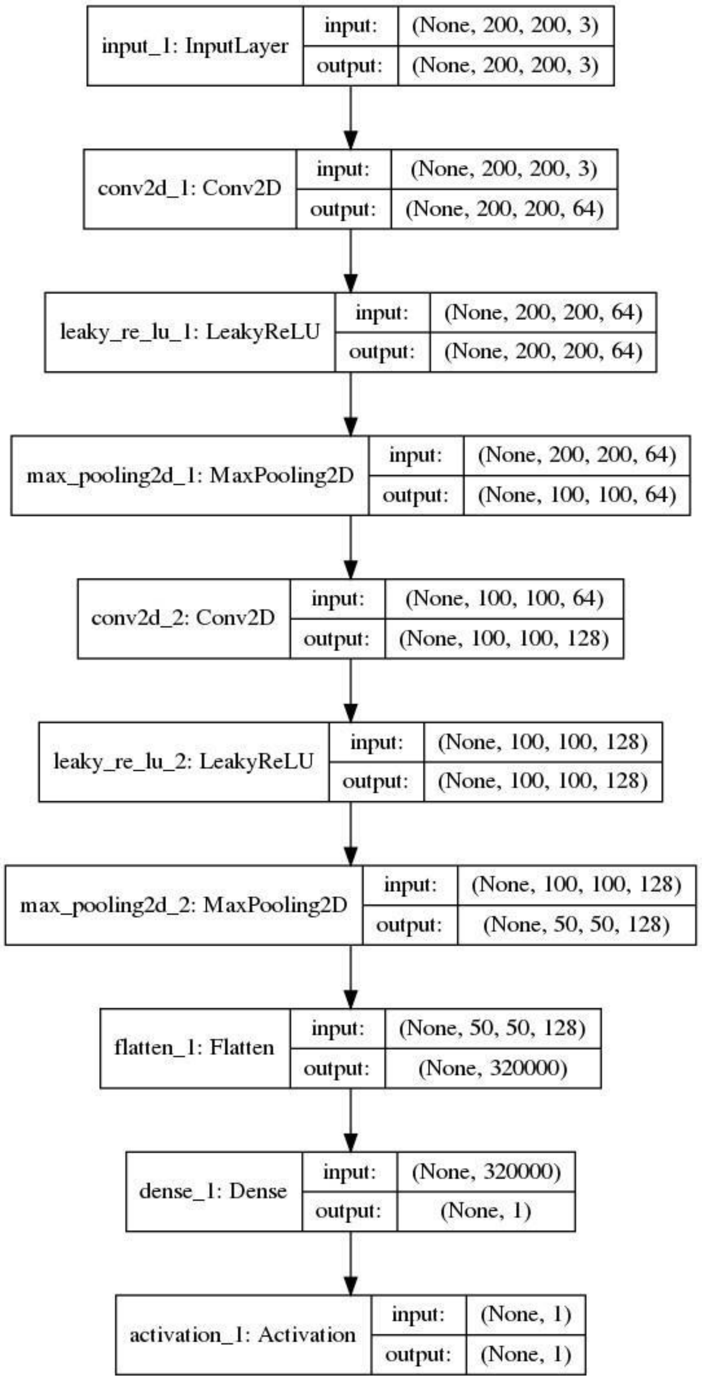

input_1: InputLayer
input: (None, 200, 200, 3)
output: (None, 200, 200, 3)
conv2d_1: Conv2D
input: (None, 200, 200, 3)
output: (None, 200, 200, 64)
leaky_re_lu_1: LeakyReLU
input: (None, 200, 200, 64)
output: (None, 200, 200, 64)
max_pooling2d_1: MaxPooling2D
input: (None, 200, 200, 64)
output: (None, 100, 100, 64)
conv2d_2: Conv2D
input: (None, 100, 100, 64)
output: (None, 100, 100, 128)
leaky_re_lu_2: LeakyReLU
input: (None, 100, 100, 128)
output: (None, 100, 100, 128)
max_pooling2d_2: MaxPooling2D
input: (None, 100, 100, 128)
output: (None, 50, 50, 128)
flatten_1: Flatten
input: (None, 50, 50, 128)
output: (None, 320000)
dense_1: Dense
input: (None, 320000)
output: (None, 1)
activation_1: Activation
input: (None, 1)
output: (None, 1)

*Fig.15: Discriminator model D architecture.*

To address this problem without making major changes to the production line and mitigating the difficulty of analysing parts under different ambient lighting conditions, this paper proposes the use of deep learning anomaly detection.

The proposed system was introduced at the end of the production line, acting during the course in which a robot handles the part. The robot was synchronized with a Basler acA3800 10gc camera, performing centralized image capture on each valve guide. Although it was necessary to change the path of the original robot, the additional triggering operation did not increase the original duty cycle time. Figure 17 shows an example of images captured from a cylinder head.

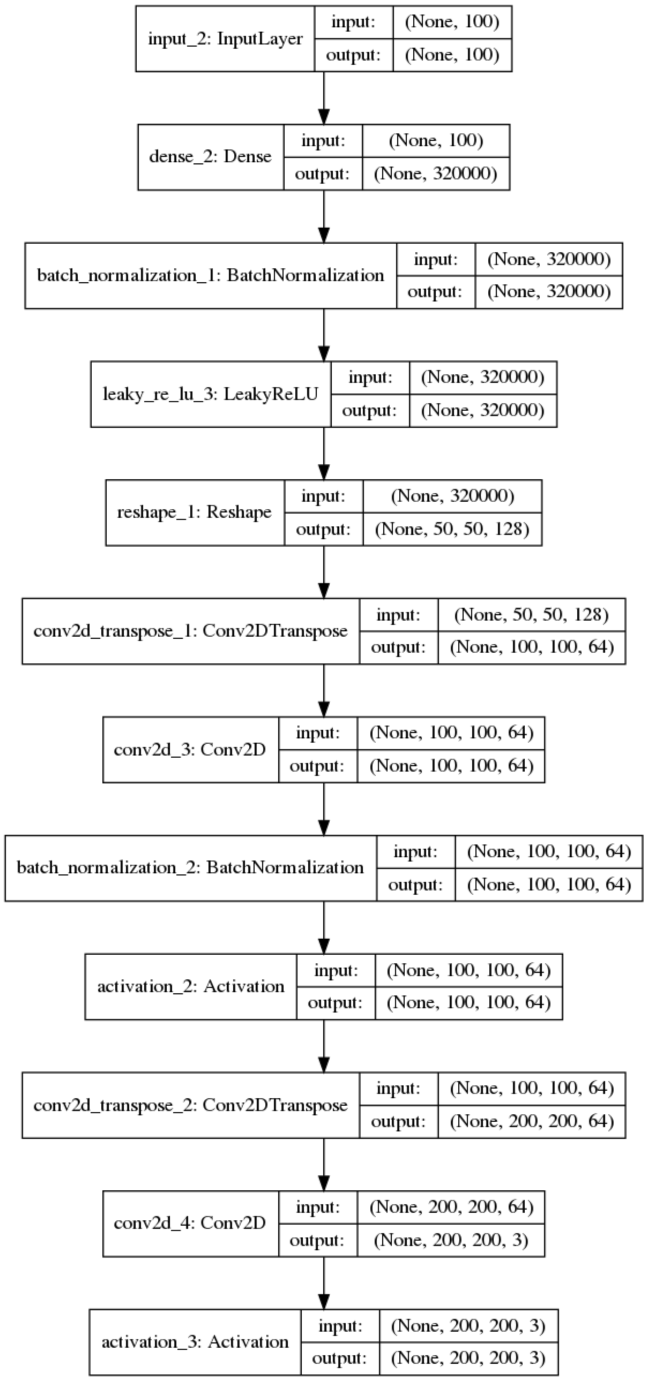

input_2: InputLayer | input: (None, 100) | output: (None, 100)
dense_2: Dense | input: (None, 100) | output: (None, 320000)
batch_normalization_1: BatchNormalization | input: (None, 320000) | output: (None, 320000)
leaky_re_lu_3: LeakyReLU | input: (None, 320000) | output: (None, 320000)
reshape_1: Reshape | input: (None, 320000) | output: (None, 50, 50, 128)
conv2d_transpose_1: Conv2DTranspose | input: (None, 50, 50, 128) | output: (None, 100, 100, 64)
conv2d_3: Conv2D | input: (None, 100, 100, 64) | output: (None, 100, 100, 64)
batch_normalization_2: BatchNormalization | input: (None, 100, 100, 64) | output: (None, 100, 100, 64)
activation_2: Activation | input: (None, 100, 100, 64) | output: (None, 100, 100, 64)
conv2d_transpose_2: Conv2DTranspose | input: (None, 100, 100, 64) | output: (None, 200, 200, 64)
conv2d_4: Conv2D | input: (None, 200, 200, 64) | output: (None, 200, 200, 3)
activation_3: Activation | input: (None, 200, 200, 3) | output: (None, 200, 200, 3)

*Fig.16: Generator model G architecture.*

The camera has been set to capture 2748 × 2748 pixels images, coloured in RGB format. To train the system, 16000 images were selected, referring to 4000 pieces considered normal according to the quality criteria. All images have been resized to 200 × 200 pixels, according to the dimensions specified in the architecture, due to the memory limitation of the GPU used. The system was trained with 10000 steps with a mini-batch of size 16. The following are the results of this case study.

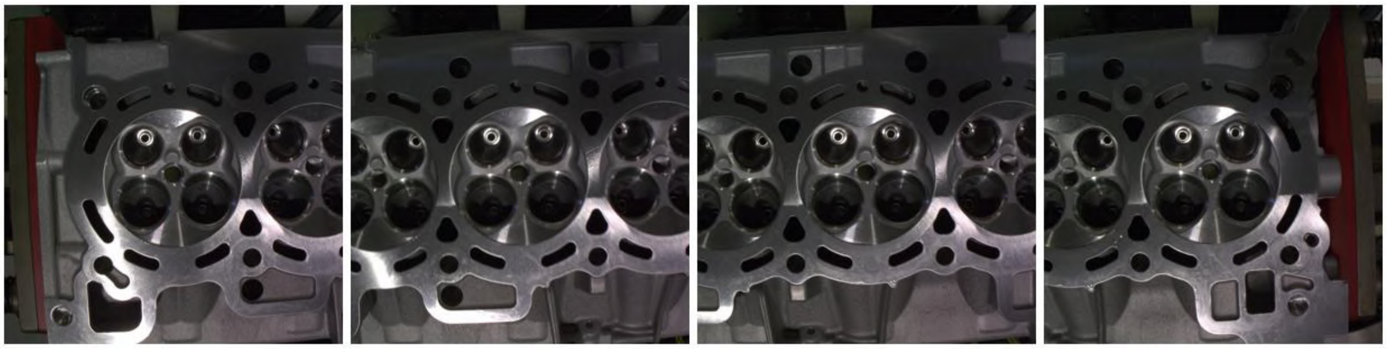


Fig.17: Examples of acquired pictures of a cylinder head.

### 3.3.3 Cylinder Head Results

To determine if the analysed image contains an anomaly, a value based on Eq. 4 was established by analysing 1000 new images considered normal, using the highest anomaly score value of the analysed images as threshold. If the input image produces a score greater than the threshold, a residual image based on Eq. 7 is generated to aid in viewing the possible anomaly region.

The test of the model in cases with anomaly occurred by evaluating real cases, such as the cases shown in Figure 18.

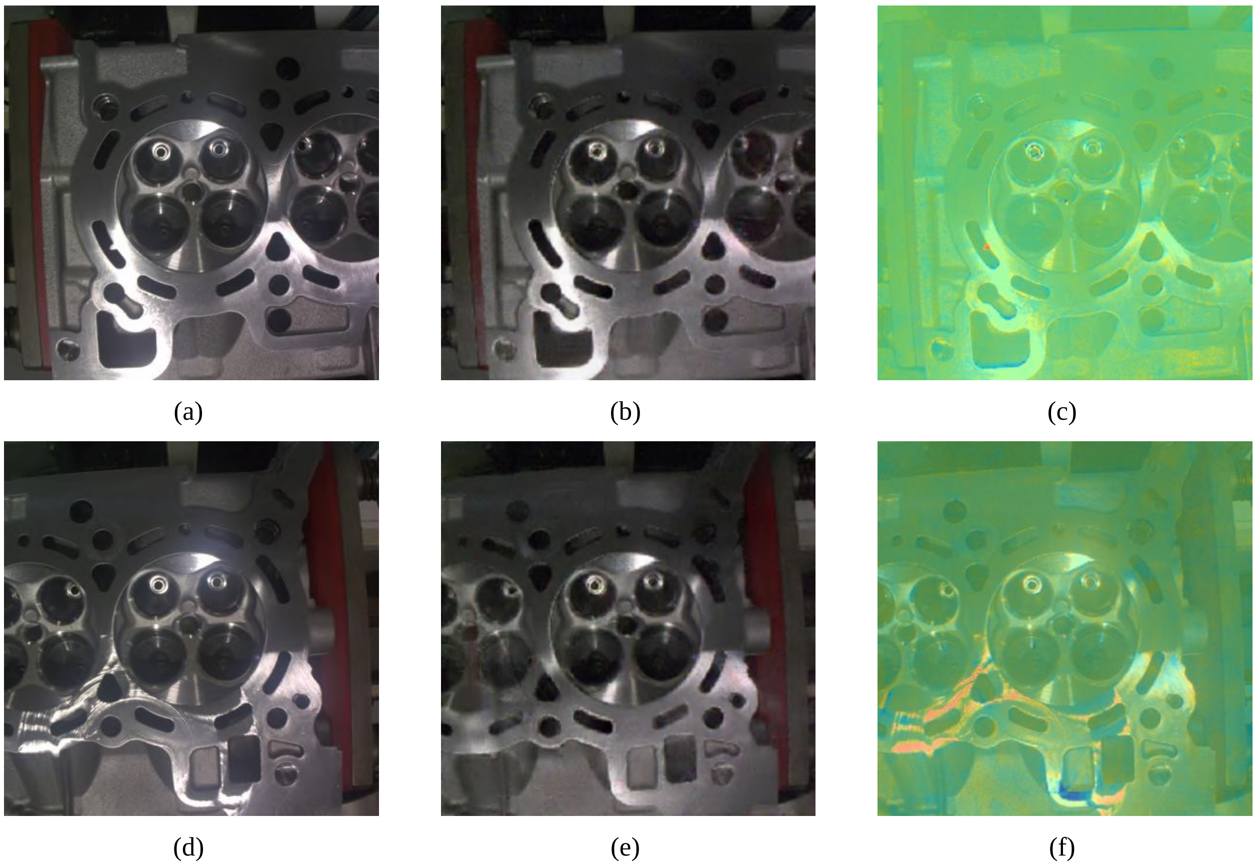


Fig.18: Anomaly detection in defective parts. (a) and (d) are the real captured images. (b) and (e) are generated image from respective input. (c) and (f) are the residual image with colormap applied over original input.

The colormap selected to this work is the colormap jet, available on opencv. The colours are represented within a range of -1 to 1, derived from the subtraction between the generated image and the original image. In this representation, the colors closest to green represent normality, whereas the closest colours to red represent anomalies.

It is important to note that what defines the anomaly with greater confidence is the value of the anomaly score, and that the colormap is only to assist the visualization of possible regions with anomaly, as described by [14].

The system is physically unable to differentiate scratches and pores smaller than 1mm, due to the scale factor of object dimensions by the amount of pixels in the image. However, by evaluating the test cases manually, it was possible to notice the limitation of the model when evaluating pores and scratches in the surface areas of the pieces smaller than 5mm$^2$.

Even with these limitations, in a side-by-side test with production line supervisors evaluating the same images, the system was able to indicate all errors pointed out. The main advantage of this system over the human operator is the evaluation time, requiring only 3 seconds per image. In addition, the system is simple to set up and economical, without having to set parameters and identify part characteristics during model training.

### 3.3.4 Brake Kit Case Study

The second case study is addressed to perform anomaly detection on the complete brake kit assembly after checking the conformity of parts. For this purpose, the location of the parts in object detection was used to cut the assembled kit region using the length between the left edge of the calliper location and the right edge of the brake disc as the width and height of cropped image. Only frames containing both object detected were used. All new images have been resized to 200 × 200 pixels for standard inputs to the architecture specified. Some examples are shown in Figure 19, related to a type 3 kit detection.

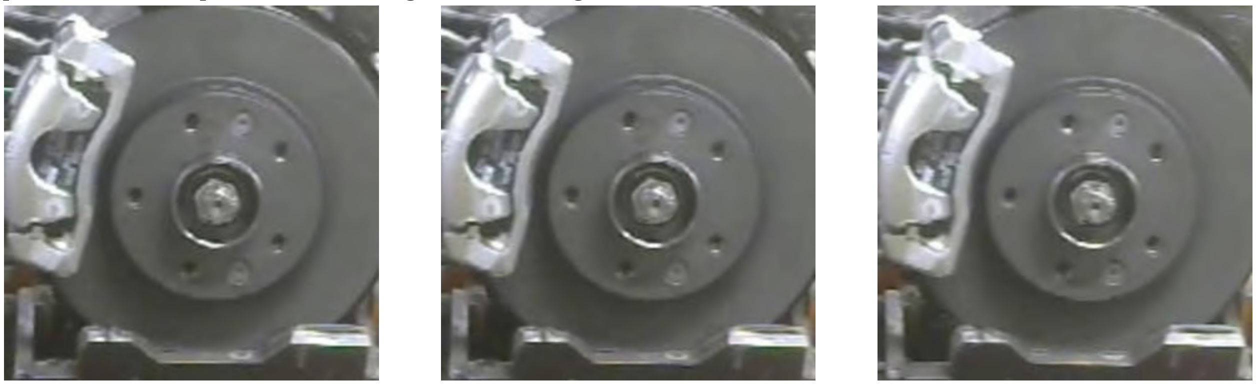


*Fig.19: Cropped frames of brake kit detection: (a) is the first detection in camera's field of view; (b) is in half of the detection path; (c) is in the last detection in camera's field of view.*

It was decided to train a specialist model for each type of brake kit, using 900 images of parts considered normal for each. The corresponding model is triggered by the result of the previous object detection, taking the frame with the highest probability detection for anomaly analysis. The parameters used for training were 1.000 steps and mini-batch of size 16.

The following subsection present the results to this case.

### 3.3.5 Brake Kit Results

To determine if the analysed image contains an anomaly, a value based on Eq. 4 was established by analysing 500 new images considered normal from each kit, using the highest anomaly score value of the analysed images as threshold. If the input image produces a score greater than the threshold, a residual image based on Eq. 7 is generated to aid in viewing the possible anomaly region.

Some examples of normal images and their analysis are shown in Figure 20, where each row concerns to a different type of brake kit. These images show the original input image, its respective image recreated by the generator model and the residual image.

The test of the model in cases with anomaly occurred by evaluating real cases captured on video, such as the case shown in Figure 21. On this occasion, the type 3 disc features a brake calliper type outside of the object trained models, inserted in the manufacturing line to a new car model. Although rejected by the conformity test in the previous step, this case was separated to evaluate the anomaly detection. In Figure 21 b it is possible to observe the image created by the generator model, trying to approximate the similar components found in the image of the components considered normal. It is important to note that what defines the anomaly with greater confidence is the value of the anomaly score, and that the colormap is only to assist the visualization of possible regions with anomaly, as described by [14].

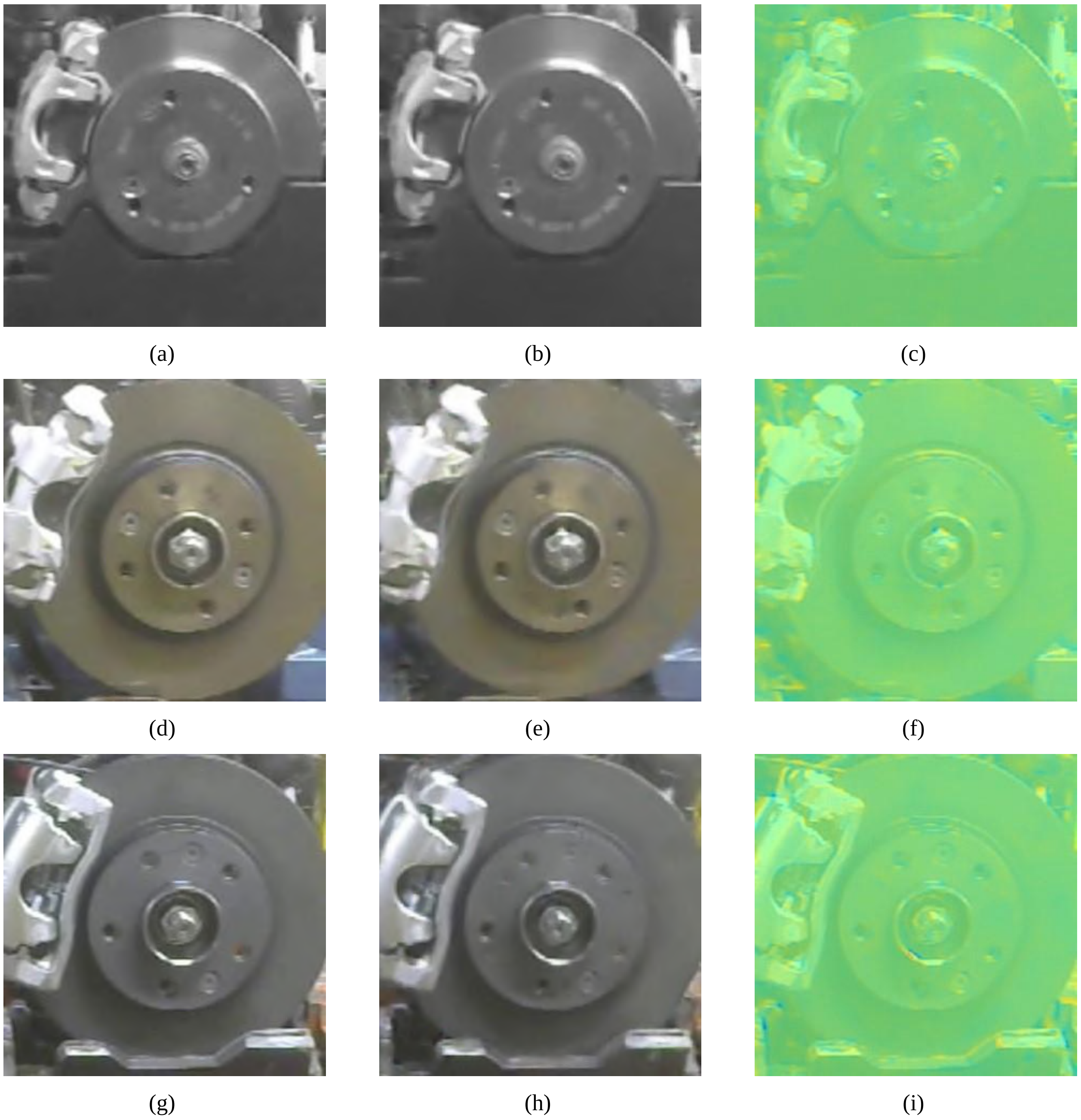


*Fig.20: Anomaly detection in normal images. (a), (d) and (g) are the real captured images. (b), (e) and (h) are generated image from respective input. (c), (f) and (i) are the residual image with colormap applied over original input.*

One contribution that the chaining of the models provides is the more detailed analysis directed to each part, since each component has a specialized model trained to detect their respective anomalies in the production process. In addition, the execution remained within the work cycle time without interfering with the analysed environment.

However, due to the camera's positioning and resolution, it was not possible to detect scratches and bumps in the anomaly detection process. These situations can be re-evaluated by changing the capture equipment used and resizing the architecture parameters such as the input dimensions and the latent space vector dimension, increasing to generate more detail.

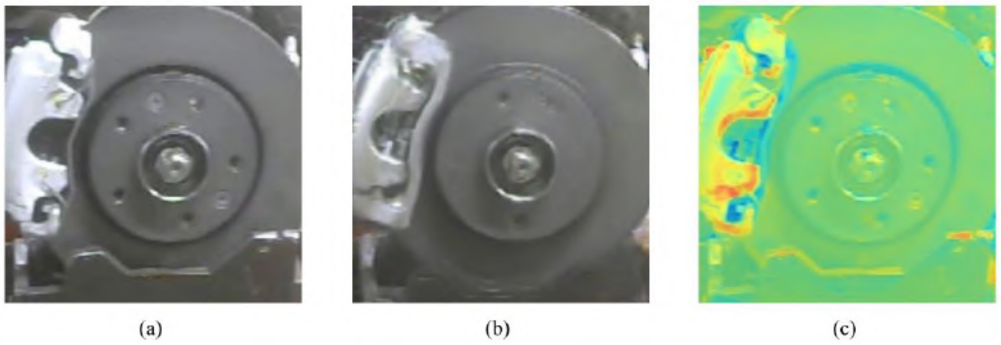


*Fig.21: Anomaly detection in brake kit with different calliper type. (a) is the real captured images. (b) is generated image from respective input. (c) is the residual image with colormap applied over original input.*

## IV. CONCLUSIONS ABOUT DEEP LEARNING APPLICATIONS IN A MANUFACTURING LINE

Based on the proofs of concept shown, it is possible to see that deep learning-based visual inspection tools can work in more generalist and less constrained environments.

As it turned out, object detection is a great tool for checking process quality, that is, evaluating the result of the operational sequence of an assembly line. The semantic segmentation tool, on the other hand, facilitates the visualization of specific regions of parts, such as a mask to evaluate areas without external interference. Finally, anomaly detection provides a robust unsupervised learning tool that can assist in product quality assessment by verifying parts at the end of production.

Moreover, the chaining of the different methodologies makes it possible to evaluate both process and product quality without interfering with the production line cycle time.

Also, being an end-to-end tool is the main feature of all these methodologies. That is, training a model can be done by a user with low understanding of machine learning or feature extraction, making it a simple categorization task for non-experts.

An impediment to these systems is the computational power required to run in real time, which requires an investment in computers with high processing power attached at the production line edge. An alternative is cloud processing systems or local servers, which in turn may exhibit higher response latency and require complex setups for installation.

Future studies should explore mechanisms for integrating production line systems with distributed deep learning [35] and other tasks for deep learning models, like content-based image retrieval with image captioning [36] and biometrics analysis [37], for operator safety purposes. Besides, other proofs of concept can be explored by applying reinforcement learning [10] to mimic the dynamic behavior of operators.

## REFERENCES

[1] O.Sami Yilmaz and Robert P. Davis. Flexible manufacturing systems: Characteristics and assessment. *Engineering Management International*, 4(3):209 – 212, 1987.

[2] Patrícia N. Pena, Tatiana A. Costa, Regiane S. Silva, and Ricardo H.C. Takahashi. Control of flexible manufacturing systems under model uncertainty using supervisory control theory and evolutionary computation schedule synthesis. *Information Sciences*, 329:491 – 502, 2016. Special issue on Discovery Science.

[3] Liangzhi Li, Kaoru Ota, and Mianxiong Dong. Deep learning for smart industry: efficient manufacture inspection system with fog computing. *IEEE Trans. on Industrial Informatics*, 14(10):4665–4673, 2018.

[4] Nirbhar Neogi, Dusmanta K. Mohanta, and Pranab K. Dutta. Review of vision-based steel surface inspection systems. *EURASIP Journal on Image and Video Processing*, 2014(1):50, Nov 2014.

[5] Mohamed El Amine Boudella, Evren Sahin, and Yves Dallery. Kitting optimisation in just-in-time mixed-model assembly lines: assigning parts to pickers in a hybrid robot-operator kitting system. *Int. Journal of Production Research*, 56(16):5475–5494, 2018.

[6] Jinjiang Wang, Yulin Ma, Laibin Zhang, Robert X. Gao, and Dazhong Wu. Deep learning for smart manufacturing: Methods and applications. *Journal of Manufacturing Systems*, 48:144 – 156, 2018.

[7] Bernd Scholz-Reiter, Daniel Weimer, and Hendrik Thamer. Automated surface inspection of cold-formed micro-parts. *CIRP Annals*, 61(1):531 – 534, 2012.

[8] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. MIT Press, Cambridge, 1 edition, 2016.

[9] Samira Pouyanfar, Saad Sadiq, Yilin Yan, Haiman Tian, Yudong Tao, Maria Presa Reyes, Mei-Ling Shyu, Shu-Ching Chen, and SS Iyengar. A survey on deep learning: Algorithms, techniques, and applications. *ACM Computing Surveys (CSUR)*, 51(5):92, 2019.

[10] Ahmed Hussein, Mohamed Medhat Gaber, Eyad Elyan, and Chrisina Jayne. Imitation learning: A survey of learning methods. *ACM Computing Surveys (CSUR)*, 50(2):21, 2017.

[11] Yoshua Bengio, Aaron C. Courville, and Pascal Vincent. Unsupervised feature learning and deep learning: A review and new perspectives. *CoRR*, abs/1206.5538, 2012.

[12] Richard Szeliski. *Computer vision: algorithms and applications*. Springer Science & Business Media, 2010.

[13] Guoqiang Zhong, Li-Na Wang, Xiao Ling, and Junyu Dong. An overview on data representation learning: From traditional feature learning to recent deep learning. *The Journal of Finance and Data Science*, 2(4):265 – 278, 2016.

[14] Thomas Schlegl, Philipp Seeböck, Sebastian M. Waldstein, Ursula Schmidt-Erfurth, and Georg Langs. Unsupervised anomaly detection with generative adversarial networks to guide marker discovery. In Marc Niethammer, Martin Styner, Stephen Aylward, Hongtu Zhu, Ipek Oguz, Pew-Thian Yap, and Dinggang Shen, editors, *Information Processing in Medical Imaging*, pages 146–157, Cham, 2017. Springer International Publishing.

[15] Y. Al-Kofahi, W. Lassoued, W. Lee, and B. Roysam. Improved automatic detection and segmentation of cell nuclei in histopathology images. *IEEE Transactions on Biomedical Engineering*, 57(4):841–852, April 2010.

[16] Holger R. Roth, Le Lu, Ari Seff, Kevin M. Cherry, Joanne Hoffman, Shijun Wang, Jiamin Liu, Evrim Turkbey, and Ronald M. Summers. A new 2.5d representation for lymph node detection using random sets of deep convolutional neural network observations. In Polina Golland, Nobuhiko Hata, Christian Barillot, Joachim Hornegger, and Robert Howe, editors, *Medical Image Computing and Computer-Assisted Intervention – MICCAI 2014*, pages 520–527, Cham, 2014. Springer International Publishing.

[17] Dumitru Erhan, Christian Szegedy, Alexander Toshev, and Dragomir Anguelov. Scalable object detection using deep neural networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2147–2154, 2014.

[18] Joseph Redmon and Ali Farhadi. Yolo9000: better, faster, stronger. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 7263–7271, 2017.

[19] Wei Liu, Dragomir Anguelov, Dumitru Erhan, Christian Szegedy, Scott Reed, Cheng-Yang Fu, and Alexander C. Berg. SSD: Single shot multibox detector. In *European Conf. on Computer Vision*, pages 21–37, 2016.

[20] Tsung-Yi Lin, Priya Goyal, Ross Girshick, Kaiming He, and Piotr Dollár. Focal loss for dense object detection. In *Proceedings of the IEEE international conference on computer vision*, pages 2980–2988, 2017.

[21] Tensorflow's object detection repository. https://bit.ly/2lPqHJk, 2019. Acessado: 23/02/2019.

[22] J. Huang, V. Rathod, C. Sun, M. Zhu, A. Korattikara, A. Fathi, I. Fischer, Z. Wojna, Y. Song, S. Guadarrama, and K. Murphy. Speed/accuracy trade-offs for modern convolutional object detectors. In *2017 IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, pages 3296–3297, 2017.

[23] Andrew G. Howard, Menglong Zhu, Bo Chen, Dmitry Kalenichenko, Weijun Wang, Tobias Weyand, Marco Andreetto, and Hartwig Adam. Mobilenets: Efficient convolutional neural networks for mobile vision applications. *arXiv preprint arXiv:1704.04861*, 2017.

[24] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014.

[25] Swarnendu Ghosh, Nibaran Das, Ishita Das, and Ujjwal Maulik. Understanding deep learning techniques for image segmentation. *ACM Computing Surveys (CSUR)*, 52(4):73, 2019.

[26] Liang-Chieh Chen, Yukun Zhu, George Papandreou, Florian Schroff, and Hartwig Adam. Encoder-decoder with atrous separable convolution for semantic image segmentation. In *ECCV*, 2018.

[27] Pierre Sermanet, David Eigen, Xiang Zhang, Michaël Mathieu, Rob Fergus, and Yann LeCun. Overfeat: Integrated recognition, localization and detection using convolutional networks. *arXiv preprint arXiv:1312.6229*, 2013.

[28] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014.

[29] Jonathan Long, Evan Shelhamer, and Trevor Darrell. Fully convolutional networks for semantic segmentation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 3431–3440, 2015.

[30] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-net: Convolutional networks for biomedical image segmentation. In *International Conference on Medical image computing and computer-assisted intervention*, pages 234–241. Springer, 2015.

[31] Vijay Badrinarayanan, Alex Kendall, and Roberto Cipolla. Segnet: A deep convolutional encoder-decoder architecture for image segmentation. *IEEE transactions on pattern analysis and machine intelligence*, 39(12):2481– 2495, 2017.

[32] Tensorflow's semantic segmentation repository. https://bit.ly/2GClDkT, 2019. Acessado: 29/09/2019.

[33] Yuan Xue, Tao Xu, Han Zhang, L Rodney Long, and Xiaolei Huang. Segan: Adversarial network with multi-scale l 1 loss for medical image segmentation. *Neuroinformatics*, 16(3-4):383–392, 2018.

[34] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. In *Advances in neural information processing systems*, pages 2672–2680, 2014.

[35] Tal Ben-Nun and Torsten Hoefler. Demystifying parallel and distributed deep learning: An in-depth concurrency analysis. *ACM Computing Surveys (CSUR)*, 52(4):65, 2019.

[36] MD Hossain, Ferdous Sohel, Mohd Fairuz Shiratuddin, and Hamid Laga. A comprehensive survey of deep learning for image captioning. *ACM Computing Surveys (CSUR)*, 51(6):118, 2019.

[37] Kalaivani Sundararajan and Damon L Woodard. Deep learning for biometrics: a survey. *ACM Computing Surveys (CSUR)*, 51(3):65, 2018.